Assignment 4

Description

In this Assignment, you need to implement four classes: `Player`, `Step`, `Game`, and `GameSystem`.

- The class `Player` represents a player and contains attributes such as player name, player ID.
- The class `Step` represents a step and contains attributes such as step ID, row index, column index, and color.
- The class `Game` represents a game and contains attributes such as game name, game ID, players from both sides, a step list, and the result board.
- The class `GameSystem` represents a game system. There is a player list and a game list in the game system. Through these lists, users can do various operations such as querying player information and querying game information.

Problems

- Problem 0: Greeting (for demo purpose) [Easy, 0 marks]
- Problem 1: Design a class named Player [Easy, 15 marks]
- Problem 2: Design a class named Step [Easy, 15 marks]
- Problem 3: Design a class named Game [Easy, 20 marks]
- Problem 4: Design a class named GameSystem [Medium, 50 marks]

Notes for this assignment

1. You should submit all source code directly into your OJ system. Do not compress them into one archive.
2. No Chinese characters are allowed to appear in your code.
3. No package statement.
4. The output must strictly follow the description and the sample in this document. Do not print anything in all methods.
5. You may need to use `String.format()` .
6. Do not call `System.exit(code)` in your code.

Problem 1: Player

Design a class named `Player` in `Player.java`

In this problem, you need to submit: `Player.java`

The `Player` describes each player. In class `Player` you need to define:

Fields:

```
private int pid
// Player ID. Each player's player ID is unique and is automatically generated by
the class Player. Starting from 1, each new player's ID will increase by 1.
```

```
private String name
// Player name.
```

```
private static int playerCnt
// Initialized to 1, increased by 1 every time a Player object is created.
```

Constructor:

```
public Player(String name)
// Constructor, automatically generate player ID and player object according to
the name provided by the user.
```

Methods:

```
public int getPid()
```

```
public String getName()
```

```
public void setName(String name)
// Modify player's name.
```

```
public static int getPlayerCnt()
// Get the value of PlayerCnt
```

```
public String toString()
// When printing the name and id for an object of this class, follow the format:
"Player: %s, pid: %d" (without quotes)
```

Note:

- Do not define a `public void setPid(int pid)` method, as Player ID should be immutable.
- You can add other methods or attributes that you think are necessary.

Problem 2: Step

Design a class named `Step` in `Step.java`

In this problem, you need to submit: `Step.java`

The `Step` describes each step. In class `Step` you need to define:

Fields:

```
private int sid
// Step ID. Step ID of each Step is unique and is automatically generated by the
class Step. Starting from 1, each new Step ID will increase by 1.
```

```
private int rowIndex
// Row index.
```

```
private int columnIndex
// Column index.
```

```
private int color
// The color of this step
```

```
private static int stepCnt
// Initialized to 1, increased by 1 every time a Step object is created.
```

Constructor:

```
public Step(int rowIndex, int columnIndex, int color)
// Constructor, automatically generate step ID and step object. Initialize row
index, column index, and color.
```

Methods:

```
public int getSid()
```

```
public int getColor()
```

```
public void setColor(int color)
```

```
public int getRowIndex()
```

```
public void setRowIndex(int rowIndex)
```

```
public int getColumnIndex()
```

```
public void setColumnIndex(int columnIndex)
```

```
public static int getStepCnt()
// Get the value of StepCnt
```

```
public String toString()
// When printing the row index, column index and color for an object of this
class, follow the format: "sid: %d, rowIndex: %d, columnIndex: %d, color: %d"
(without quotes)
```

Note:

- Do not define a `public void setSid(int sid)` method, as Step ID should be immutable.
- You can add other methods or attributes that you think are necessary.

Problem 3: Game

Design a class named `Game` in `Game.java`

In this problem, you need to submit: `Game.java`

The `Game` describes each game. In class `Game` you need to define:

Fields:

```
private int gid
// Game ID. Each game's game ID is unique and is automatically generated by class
Game. Starting from 1, each new game's ID will increase by 1.
```

```
private String name
// Game name.
```

```
private static int gameCnt
// Initialized to 1, increased by 1 every time a Game object is created
```

```
private Player whitePlayer
// The white player object corresponding to this game.
```

```
private Player blackPlayer
// The black player object corresponding to this game.
```

```
private Arraylist<Step> stepList
// The list of steps.
```

```
private int[][] board
// The board of the game. The size of the board is 8x8.
```

Constructor:

```
public Game(String name, Player whitePlayer, Player blackPlayer)
// Constructor, automatically generate game ID and Game object according to the
name provided by the user. Initialize white player, black player, stepList, and
board.
// For a list with no elements in it, its size should be 0 and its reference
should not be null.
// For a board that has not to be set:
//  1. Its size should be 8*8.
//  2. board[3][3] and board[4][4] should be white(1). board[3][4] and board[4][3]
should be black(-1)
//  3. Its reference should not be null.
```

Methods:

```
public int getGid()
```

```
public String getName()
```

```
public void setName(String name)
// Modify game name.
```

```
public static int getGameCnt()
```

```
public Player getWhitePlayer()
```

```
public Player getBlackPlayer()
```

```
public ArrayList<Step> getStepList()
```

```
public boolean checkStep(int sid)
// If a step is not in the step list, return false.
// Otherwise return true.
```

```
public boolean addStep(Step step)
// If a step is in the step list (i.e. The sid is in the step list), return false.
// Otherwise add the step into the step list and return true.
```

```
public int[][] getBoard()
```

```
public void setBoard(int[][] board)
// Change the game result
```

```
public String toString()
// When print an object of this class, follow the format: "Game: %s, gid: %d,
whitePlayerId: %d, blackPlayerId: %d, stepList: %s, board: %s"
// 1. The stepList String should be in the format: "[sid: x, rowIndex: x,
columnIndex: x, color: x, sid: x, rowIndex: x, columnIndex: x, color: x, ...]"
// 2. The board string should be in the format: "[[x, x, x, x, x, x, x, x], [x, x,
x, x, x, x, x, x], ...]"
// 3. Without quotes
```

Note:

- Do not define these setters:

  ```
  public void setGid(int gid)
  ```

  ```
  public void setWhitePlayer(Player whitePlayer)
  ```

  ```
  public void setBlackPlayer(Player blackPlayer)
  ```

- You may need to use `Arrays.deepToString()` for board string .
- You can add other methods or attributes that you think are necessary.

Problem 4: GameSystem

Design a class named `GameSystem` in `GameSystem.java`

In this problem, you need to submit: `Player.java`, `Step.java`, `Game.java`, `GameSystem.java`

`GameSystem` describes each game and player. In class `GameSystem` you need to define:

Fields:

```
private ArrayList<Player> playerList
// The list of players
```

```
private ArrayList<Game> gameList
// The list of games
```

Constructor:

```
public GameSystem()
// Constructor. Initialize playerList and gameList. For a list with no elements in it, its size should be 0 and its reference should not be null.
```

Methods:

```
public ArrayList<Game> getGameList()
```

```
public ArrayList<Player> getPlayerList()
```

```
public boolean checkPlayer(int pid)
// If the player with pid is not in the player list, return false.
// Otherwise return true.
```

```
public boolean checkGame(int gid)
// If the game with gid is not in the game list, return false.
// Otherwise return true.
```

```
public boolean addPlayer(Player player)
// If the player is in the player list (i.e. his/her pid is in the player list), return false.
// Otherwise add the player into the player list and return true.
```

```
public boolean addGame(Game game)
// If the game is in the game list (i.e. its gid is in the game list), return false.
// Else if the corresponding players of the game is not in playerList, return false.
// Otherwise add the game into the game list and return true.
```

```
public ArrayList<Game> listPlayerGame(int pid)
// Return an ArrayList<Game> of the player with pid, the order of the games is following the game order in the gameList. Skip games the player didn't participate in.
```

```
public float calculatePlayerWinRate(int pid)
// Return the win rate of the player with pid. The win rate is the ratio of total
wins to total games the player has participated in.
// If the player has not participated in any games, return 0.
```

Notes:

- You can add other methods or attributes that you think are necessary.
- Do not sort the lists.
- To simplify the problem, we use numbers -1, 0, 1 to represent black disc, blank, and white disc.